#3

Docket No.: 50023-165 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	
Yuko KUBOOKA, et al.	:	
Serial No.:	:	Group Art Unit:
Filed: January 22, 2002	:	Examiner:

For: DIGITAL DEVICE, TASK MANAGEMENT METHOD AND PROGRAM THEREFOR

CLAIM OF PRIORITY AND
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Commissioner for Patents
Washington, DC 20231

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicants hereby claim the priority of:

Japanese Patent Application No. 2001-016601, filed January 25, 2001

cited in the Declaration of the present application. A Certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

Stephen A. Becker
Registration No. 26,527

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 SAB:prp
Date: January 22, 2002
Facsimile: (202) 756-8087

Kubooka et al.
January 22, 2002
McDermott, Will & Emery

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００１年 １月２５日

出願番号
Application Number: 特願２００１－０１６６０１

出願人
Applicant(s): 松下電器産業株式会社

CERTIFIED COPY OF
PRIORITY DOCUMENT

２００１年１１月２６日

特許庁長官
Commissioner,
Japan Patent Office

及川耕造

出証番号 出証特２００１－３１０２９６４

【書類名】　特許願
【整理番号】　2022530017
【提出日】　平成13年 1月25日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F　9/46
【発明者】
　【住所又は居所】　東広島市鏡山３丁目１０番１８号　株式会社松下電器情報システム広島研究所内
　【氏名】　久保岡　祐子
【発明者】
　【住所又は居所】　東広島市鏡山３丁目１０番１８号　株式会社松下電器情報システム広島研究所内
　【氏名】　土井　繁則
【特許出願人】
　【識別番号】　000005821
　【氏名又は名称】　松下電器産業株式会社
【代理人】
　【識別番号】　100083172
　【弁理士】
　【氏名又は名称】　福井　豊明
【手数料の表示】
　【予納台帳番号】　009483
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　9713946
【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　デジタル機器、タスク管理方法及びそのプログラム

【特許請求の範囲】

【請求項１】　一実行単位として管理されるタスクを、プログラム実行手段にて複数並列に実行するオペレーティングシステムを備えたデジタル機器において、

上記タスクを構成する関数の属性を記憶するタスク属性情報記憶手段と、

実行されている関数の属性に関する情報をタスク属性認識手段に送信すると共に、当該関数の属するタスクに対する強制終了の実行をタスク実行判断手段に問い合わせる上記プログラム実行手段と、

上記強制終了の実行の問い合わせに対し、少なくとも実行されている関数の属性に基づいて上記強制終了の実行を判定するタスク実行判断手段と、

上記プログラム実行手段から送信された上記関数の属性に関する情報に基づいて、実行されている関数の属性を上記タスク属性情報記憶手段に格納すると共に、上記タスク実行手段からの実行されている関数の属性の問い合わせに対して、上記タスク属性情報記憶手段に格納した上記関数の属性を返信するタスク属性認識手段とを具備することを特徴とするデジタル機器。

【請求項２】　上記関数の属性に関する情報が、実行されている関数の名称であると共に、

上記タスク属性認識手段は、上記関数のコードと、上記名称に基づいて上記実行されている関数の属性を判定する請求項１に記載のデジタル機器。

【請求項３】　上記一実行単位として管理されるタスクは、プロセスである請求項１に記載のデジタル機器。

【請求項４】　上記一実行単位として管理されるタスクは、スレッドである請求項１に記載のデジタル機器。

【請求項５】　一実行単位として管理されるタスクを、複数並列に実行するオペレーティングシステムにおけるタスク管理方法において、

上記タスクを構成する関数であって、現在実行されている関数の属性を記憶し、

上記実行されている関数の属するタスクに対する強制終了の実行時に、上記記憶した関数の属性に基づいて、上記実行されている関数の強制終了の可否を判定することを特徴とするタスク管理方法。

【請求項６】　上記関数の属性が、実行されている関数の名称を用いて決定される請求項５に記載のタスク管理方法。

【請求項７】　上記一実行単位として管理されるタスクは、プロセスである請求項５に記載のタスク管理方法。

【請求項８】　上記一実行単位として管理されるタスクは、スレッドである請求項５に記載のタスク管理方法。

【請求項９】　一実行単位として管理されるタスクを、プログラム実行機能にて複数並列に実行するオペレーティングシステムをコンピュータに実現させるプログラムにおいて、

実行されている関数の属性に関する情報をタスク属性認識機能に送信すると共に、当該関数の属するタスクに対する強制終了の実行をタスク実行判断機能に問い合わせる上記プログラム実行機能と、

上記強制終了の実行の問い合わせに対し、少なくとも実行されている関数の属性に基づいて上記強制終了の実行を判定するタスク実行判断機能と、

上記プログラム実行機能から送信された上記関数の属性に関する情報に基づいて、実行されている関数の属性を記憶手段に格納すると共に、上記タスク実行機能からの実行されている関数の属性の問い合わせに対して、上記記憶手段に格納した上記関数の属性を返信するタスク属性認識機能とをコンピュータに実現させるためのプログラム。

【請求項１０】　一実行単位として管理されるタスクを、プログラム実行機能にて複数並列に実行するオペレーティングシステムをコンピュータに実現させるプログラムにおいて、

実行されている関数の属性に関する情報をタスク属性認識機能に送信すると共に、当該関数の属するタスクに対する強制終了の実行をタスク実行判断機能に問い合わせる上記プログラム実行機能と、

上記強制終了の実行の問い合わせに対し、少なくとも実行されている関数の属

性に基づいて上記強制終了の実行を判定するタスク実行判断機能と、

上記プログラム実行機能から送信された上記関数の属性に関する情報に基づいて、実行されている関数の属性を記憶手段に格納すると共に、上記タスク実行機能からの実行されている関数の属性の問い合わせに対して、上記記憶手段に格納した上記関数の属性を返信するタスク属性認識機能とを実現させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項１１】　一実行単位として管理されるタスクを、プログラム実行手段にて複数並列に実行するオペレーティングシステムを備えたデジタル機器において、

上記タスクの属性を記憶するタスク属性情報記憶手段と、

上記タスクの属性に関する情報をタスク属性認識手段に送信すると共に、当該タスクに対する強制終了の実行をタスク属性判断手段問い合わせる上記プログラム実行手段と、

上記強制終了の実行の問い合わせに対し、少なくともタスクの属性に基づいて上記強制終了の実行を判定するタスク実行判断手段と、

上記プログラム実行手段から送信された上記タスクの属性に関する情報に基づいて、当該タスクの属性を上記タスク属性情報記憶手段に格納すると共に、上記タスク実行手段からのタスクの属性の問い合わせに対して、上記タスク属性情報記憶手段に格納した上記タスクの属性を返信するタスク属性認識手段とを具備することを特徴とするデジタル機器。

【請求項１２】　上記一実行単位として管理されるタスクは、プロセスである請求項１１に記載のデジタル機器。

【請求項１３】　上記一実行単位として管理されるタスクは、スレッドである請求項１１に記載のデジタル機器。

【請求項１４】　一実行単位として管理されるタスクを、複数並列に実行するオペレーティングシステムにおけるタスク管理方法において、

実行されている上記タスクの属性を記憶し、

上記タスクに対する強制終了の実行時に、上記記憶したタスクの属性に基づいて、上記実行されているタスクの強制終了の可否を判定することを特徴とするタ

スク管理方法。

【請求項１５】　上記一実行単位として管理されるタスクは、プロセスである請求項１４に記載のタスク管理方法。

【請求項１６】　上記一実行単位として管理されるタスクは、スレッドである請求項１４に記載のタスク管理方法。

【請求項１７】　一実行単位として管理されるタスクを、プログラム実行機能にて複数並列に実行するオペレーティングシステムをコンピュータに実現させるプログラムにおいて、

実行されているタスクの属性に関する情報をタスク属性認識機能に送信すると共に、当該タスクに対する強制終了の実行をタスク実行判断機能に問い合わせる上記プログラム実行機能と、

上記強制終了の実行の問い合わせに対し、少なくとも実行されているタスクの属性に基づいて上記強制終了の実行を判定するタスク実行判断機能と、

上記プログラム実行機能から送信された上記タスクの属性に関する情報に基づいて、実行されているタスクの属性を記憶手段に格納すると共に、上記タスク実行機能からの実行されているタスクの属性の問い合わせに対して、上記記憶手段に格納した上記タスクの属性を返信するタスク属性認識機能とをコンピュータに実現させるためのプログラム。

【請求項１８】　一実行単位として管理されるタスクを、プログラム実行機能にて複数並列に実行するオペレーティングシステムをコンピュータに実現させるプログラムにおいて、

実行されているタスクの属性に関する情報をタスク属性認識機能に送信すると共に、当該タスクに対する強制終了の実行をタスク実行判断機能に問い合わせる上記プログラム実行機能と、

上記強制終了の実行の問い合わせに対し、少なくとも実行されているタスクの属性に基づいて上記強制終了の実行を判定するタスク実行判断機能と、

上記プログラム実行機能から送信された上記タスクの属性に関する情報に基づいて、実行されているタスクの属性を記憶手段に格納すると共に、上記タスク実行機能からの実行されているタスクの属性の問い合わせに対して、上記記憶手段

に格納した上記タスクの属性を返信するタスク属性認識機能とを実現させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、デジタル機器、タスク管理方法及びそのプログラムに関し、詳しくは、マルチタスクオペレーションシステムにてタスクの強制終了を管理するデジタル機器、タスク管理方法及びそのプログラムに関する。

【０００２】

【従来の技術】

従来では、例えばＳＴＢ（set-top box：ケーブルTVのコントロールボックス等、家庭用テレビに接続して追加機能を提供するデバイスの一般名称）や携帯電話等、主にソフトウェアを用いて様々な機能を実現する製品の場合、当該ソフトウェアは上記製品に、ＯＳ（オペレーションシステム）及び各機能を実現するプログラムを予め格納した、いわゆる組込ＯＳという形で提供される。上記組込ＯＳを備えた製品を提供するメーカーは、当該組込ＯＳの各機能の動作を十分に検証し、様々な動作状態におけるデバッグを行うことができるため、安定動作する信頼性の高い製品を提供することが可能である。

【０００３】

又、近年では、製品によってはライフタイムが短く、又新機能の追加・変更サイクルも短くなっている。このため、当該機能追加・変更に柔軟に対応するべく、上記組込ＯＳに替えて汎用ＯＳを用いるケースが増えている。さらにネットワークの発達（多様化）やＪａｖａに代表されるネットワークに適した多機能なインタプリタ言語の普及により、機能追加・変更に伴うプログラムのダウンロードが容易になってきている点も、上記組込ＯＳの汎用ＯＳ化を促進する要因となっている。

【０００４】

以下に上記汎用ＯＳを用いた際の一般的なタスクの実行形態を説明する。尚、上述したような汎用ＯＳは通常マルチタスクとして動作するため、該マルチタス

クを含めた説明を行う。

【０００５】

マルチタスクＯＳでは、図５（ａ）に示すように、ＯＳ５０１上で例えば複数のプロセス１、２（５０２、５０３）が実行される。又、上記プロセス１（５０２）の管理下にて、さらに例えばスレッド１（５０４）、スレッド２（５０５）、上記プロセス２（５０３）の管理下では例えばスレッド４（５０６）という様に、複数のスレッドが実行される。尚、例えば上記プロセス１（５０２）は、実行時に上記ＯＳ５０１よりメモリ空間やＩ／Ｏ（Input／Output）空間等のリソース（資源）が割り当てられ、当該リソースは上記スレッド１（５０４）及びスレッド２（５０５）にて共有される。

【０００６】

ここで、上記プロセスとは、プログラムの一実行単位を指す。即ち、上記リソース等を独立して所有すると共に、切り替え時（マルチタスクＯＳによるプロセスの切り替え時）においてＣＰＵレジスタの内容をすべて保存し、次に制御を切り替えるプロセスのためのレジスタ値をロードすることにより各実行単位の独立を保障している単位である。又、上記スレッドもプログラムの一実行単位であり、同一プロセス内でのマルチタスク処理を可能にしたものである。尚、上記プロセスやスレッドという単位は、ＯＳにより扱いが異なるが本質はプログラムの一実行単位であり、関数の集合ということができる。このため以降、適宜プロセスとスレッドを使い分けて説明するが、双方ともタスクであり、プログラムの一実行単位である。さらに、アプリケーションも広義のタスクである。

【０００７】

次に、上述したタスク（プロセス及びスレッド）の構造について簡単に説明する。上記タスクは一般的に、関数の集合により構成されており、例えば所定の関数が別の関数に引数を渡し、当該別の関数が計算した戻り値を用いて次の処理を実行するといった処理が行われる。例えば図６を用いて説明すると、タスク（スレッド１）６０１は、ユーザが開発したアプリケーション関数ａ（６０２）をメインルーチン（メイン関数）とし、当該メインルーチンの処理の過程で、アプリケーション関数ｂ（６０３）を呼び出している。当該アプリケーション関数ｂ（

６０３）は、さらに必要に応じてシステムライブラリ関数として提供されるシステムライブラリ関数ａ（６０４）を読み出し、さらに当該システムライブラリ関数ａ（６０４）はアプリケーション関数ｃ（６０５）を呼び出している。上記処理の過程で呼び出された関数は、それぞれ呼び出し時に与えられた引数を用いて戻り値を返し、それに応じて呼び出した関数は続く処理を実行する。

【０００８】

以上のように、タスクを構成する複数の関数がそれぞれ処理されることにより、所定のタスク（メインルーチン）の処理が完了する。ここに、上記アプリケーション関数とは、例えば当該タスクをプログラミングした者が必要に応じて作成した関数であり、システムライブラリ関数とは、予め提供されている基本的な関数等である。尚、上記アプリケーション関数及びシステムライブラリ関数共、例えば数十から数千のステップ（命令）にて構成されている。

【０００９】

さて、上述した各スレッド１（５０４）及びスレッド２（５０５）は、図５（ｂ）に示すように、上記ＯＳ５０１にて順次割り当てられた処理時間において実行される。即ち、ＯＳ５０１は、タイマー処理にてスレッド１とスレッド２（ここではスレッド３の処理は考慮しないものとする）それぞれに処理時間５０７～５１０を与える。ここでは、例えば、まず処理時間５０７がＯＳ５０１により割り当てられた時にスレッド１が実行され、続いて割り当てられる処理時間５０８には、スレッド２が実行される。このように、ＯＳが処理時間を複数のスレッド（タスク）に順次与えることにより、複数の処理を並列に実行することが可能となり、即ちマルチタスクＯＳを実現可能となる。尚、上記スレッド（タスク）の切り替えをディスパッチという。

【００１０】

以上に説明した汎用ＯＳを用い、さらにネットワークの発達（多様化）や多機能なインタプリタ言語の普及等により、様々な製品への新たな機能追加等を柔軟に行うことが可能になっている。

【００１１】

尚、新たに機能追加等によるシステムの変更後には、当該新機能を組み合わせ

た様々な使用形態が考えられる。このような状態では、通常、リソースが大量に消費されてしまい、後に実行されるプログラムにリソースを割り当てることができず、このため新たなプログラムを起動できなくなる。このような状態を回避するために、従来では、例えば他のタスク（プロセスやスレッド）の処理を終了させる命令、即ち強制終了命令を発行することが可能である。当該強制終了命令により、異なるタスクのリソースを開放し、新たなプログラムを起動することが可能となる。上記リソースの不足は、ある程度機能が限定されている製品（例えば上記ＳＴＢ、携帯電話等）においては、リソースを十分に備えていないことが多く、汎用のパーソナルコンピュータとは異なり起こりやすいといえる。

【００１２】

但し、むやみに他のタスクの処理を強制終了可能とすると、当該他のタスクがデータの書き込み等の重要な処理を行いっている最中に終了してしまうといったことが考えられる。このため、特開平１０－６９３９２に記載の技術では、所定の処理区間に対して「強制終了禁止区間（アボート禁止区間）」を設定可能とすることにより当該処理区間における強制終了の禁止を可能としている。

【００１３】

以下図７を用いて、タスクの処理を終了させる命令の発行の仕組み及び、強制終了禁止区間の設定について説明する。

【００１４】

図７において、スレッド１（６０１）、及びスレッド２（７０１）がＯＳ上にて独立したタスクとして起動しているものとする。尚、上記スレッド１（６０１）及びスレッド２（７０１）は、上述したマルチタスク処理にて、随時切り替えられながら処理を実行しており、例えばスレッド１（６０１）に割り当てられた処理時間が終了すると、次にスレッド２（７０１）が動作する。

【００１５】

ここで、スレッド１（６０１）におけるシステムライブラリ関数ａ（６０４）の実行時７０２に処理が上記スレッド２（７０１）に切り替えられた場合、スレッド２（７０１）は例えばアプリケーション関数ｄ（７０３）から処理を実行する。

【００１６】

続いてスレッド２（７０１）におけるアプリケーション関数ｅ（７０４）の処理の所定のポイント７０５にて、スレッド２（７０１）が、スレッド１（６０１）に強制終了命令を発行し、アプリケーション関数ｄ（７０３）の所定のポイント７０６にてスレッド１（６０１）に処理が切り替わったと仮定する。この場合には、例えばスレッド１（６０１）におけるシステムライブラリ関数ａ（６０４）実行時に強制終了命令を受け付けることになり、当該システムライブラリ関数ａ（６０４）実行中（上記実行時７０２直後）にスレッド１（６０１）が例えばＯＳにより強制終了されることになる。尚、上記各関数（６０２～６０５）による処理が強制終了されると、全体の処理に大きな影響を与える場合等には以下のような強制終了禁止区間が設けられる。

【００１７】

即ち、スレッド１（６０１）におけるアプリケーション関数ａ（６０２）の所定のポイント７０７にて、当該アプリケーション関数ａ（６０２）は強制終了禁止命令７０８を発行する。当該強制終了禁止命令は、例えばＯＳにおける所定の処理にて認識され、記憶される。次に、上記同様、スレッド２（７０１）から所定のポイント７０５で強制終了命令が発行され、処理がスレッド１（６０１）に切り替えられると、例えばＯＳが、上記強制終了禁止命令が発行されているか否かを判定し、ここでは強制終了禁止命令が発行されているため強制終了せずに通常の処理を継続する。

【００１８】

続いてアプリケーション関数ａ（６０２）が所定のポイント７０９にて、強制終了許可命令７１０を発行すると、当該強制終了許可命令は例えばＯＳにおける所定の処理にて認識され、強制終了禁止命令７０８発行時から強制終了許可命令７１０の発行時までの区間、即ち、強制終了禁止区間が終了する。

【００１９】

上記強制終了禁止区間が終了すると、例えばＯＳは、上記所定のポイント７０５にてスレッド２（７０１）により発行された強制終了命令を反映し、アプリケーション関数ａ（６０２）における所定のポイント７０９にてスレッド１（６０

１）の強制終了を行う。

【００２０】

以上により、タスク間（スレッド間）にて強制命令を発行することで、例えば他のタスクが使用しているリソースを解放させることが可能となる。又、システムに影響を与えるような重要な処理（例えば共有リソースに対する操作）を行っている場合には、他のタスクによる強制終了を禁止することで、当該重要な処理の正常終了を可能としている。

【００２１】

【発明が解決しようとする課題】

しかしながら、上記従来技術を用いた場合には以下のような問題が生じる。即ち、上述したように、ネットワークを用いて例えば新機能を備えたプログラム（ソフトウェア）を汎用ＯＳにダウンロードして実行した場合には、組込ＯＳと異なり、各機器により異なる複数のプログラムが実行されている等、ユーザー環境が多岐に渡るため正常動作しないことが考えられる。このような場合には、上記プログラムを強制終了させる必要があるが、上述した強制終了禁止区間が設定されてしまうと、強制終了することができないといった問題が起こる。

【００２２】

又、特にインターネットを介して第三者が作成したプログラムをダウンロードする場合には、当該プログラムは必ずしも善意の基に作成されたものとは限らないため、当該プログラムを実行するには危険が伴う。

【００２３】

即ち、例えば図７に示したスレッド１（６０１）が悪意を持ったプログラムであった場合、具体的には、例えばアプリケーション関数ｂ（６０３）内にて無限ループし、かつ同様のアプリケーション関数ｂ（６０３）を持ったスレッドを生成するプログラムであった場合が考えられる。このような場合には、上記悪意を持ったプログラムを実行してしまうと、スレッドが増殖し、さらに当該スレッドに対する強制終了命令も、上述した強制終了禁止区間の設定により無効になってしまう。こうなると、上記プログラムを実行した端末（製品）は、ＯＳの制御が効かなくなり、最終的には電源のオフ等が必要になってしまう。

【００２４】

このような現象は、汎用ＯＳを用いて特定の機能を提供する製品に限らず、一般的な汎用ＯＳであるＷｉｎｄｏｗｓやＵＮＩＸでも考えられることである。

【００２５】

そこで本発明は、上記従来の事情に基づいて提案されたものであって、システムに障害をもたらす可能性のあるタスク、例えば共有リソースの操作中には、当該タスクを強制終了させないタスク管理方法を提供することを目的とする。

【００２６】

【課題を解決するための手段】

本発明は、上記目的を達成するために以下の手段を採用している。すなわち、本発明は、一実行単位として管理されるタスクを、プログラム実行手段にて複数並列に実行するオペレーティングシステムを備えたデジタル機器を前提としている。ここで、プログラム実行手段は、実行されている関数の属性に関する情報をタスク属性認識手段に送信すると共に、当該関数の属するタスクに対する強制終了の実行をタスク実行判断手段に問い合わせる。又、タスク事項判断手段は、上記強制終了の実行の問い合わせに対し、少なくとも実行されている関数の属性に基づいて上記強制終了の実行を判定する。さらに、タスク属性認識手段は、プログラム実行手段から送信された上記関数の属性に関する情報に基づいて、実行されている関数の属性をタスク属性情報記憶手段に格納すると共に、上記タスク実行手段からの実行されている関数の属性の問い合わせに対して、上記タスク属性情報記憶手段に格納した上記関数の属性を返信する。

【００２７】

以上の構成では、関数の呼び出し時、及び処理が戻った時に、タスクがタスク属性認識手段に以後実行される関数の種類を通知することで、タスク属性情報記憶手段には、現在実行されているタスクの種類が常時格納され、このため、タスク属性情報記憶手段に格納されている、現在のタスクにて実行されている関数の属性情報を参照・判定可能となる。従って、強制終了禁止区間であっても、所定の関数を実行している場合には、強制終了可能とし、一方でシステムに障害をもたらす可能性のある別の関数の実行時には、当該タスクを強制終了させないとい

った強制終了についての柔軟な制御が可能である。

【００２８】

又、上記関数の属性に関する情報を実行されている関数の名称とし、タスク属性認識手段が、上記関数のコードと、上記名称に基づいて上記実行されている関数の属性を判定する構成もある。

【００２９】

この構成では、言語によっては、当該言語の仕様により関数の属性を判断し、直接上記タスク属性認識手段に通知することが困難な場合があが、このような場合でも実行している関数の属性を確認することができる。

【００３０】

尚、上記一実行単位であるタスクは、プロセスである場合や、スレッドである場合がある。

【００３１】

さらに、タスク属性情報記憶手段が、タスクを構成する関数の属性に代えてタスク自体の属性を格納することにより、タスクの属性に基づいた強制終了の制御を可能とすることができる。

【００３２】

尚、プログラム及びプログラム記録媒体は、コンピュータにて上記各手段の機能を実現する構成である。

【００３３】

【発明の実施の形態】

以下、添付図面を参照して、本発明の実施の形態につき説明し、本発明の理解に供する。尚、以下の実施の形態は、本発明を具体化した一例であって、本発明の技術的範囲を限定する性格のものではない。

【００３４】

〔実施の形態１〕

本発明の実施の形態１におけるデジタル機器１００について図１、図２、図４を参照し説明する。ここに、上記デジタル機器１００とは、例えばＳＴＢ、携帯電話、家電、パーソナルコンピュータ等であって、外部からのアプリケーション

（プログラム）を付加（追加）等することにより、当該デジタル機器への所定機能の追加・変更・修正が可能である機器を示す。

【００３５】

まず、上記デジタル機器１００は、例えばインターネット等のネットワーク１０１を介して所定のアプリケーション（プログラム：１０２）を取得し、記憶手段１０３に記憶する。ここに、当該アプリケーション１０２は、例えば上記デジタル機器１００に新たな機能を追加するためのインタプリタ言語にて記述されたプログラムであり、実行時には、図２に示すタスク２０１を生成して処理を実行するものとする。尚、上記アプリケーション１０２は、必ずしもネットワーク１０１を介する必要は無く、メモリーカード等の記憶媒体を介して得るものであってもよい。

【００３６】

次に、上記アプリケーション１０２は、実行時には例えばローダー等により、ＲＡＭ（Random Access Memory）等の一時記憶手段１０５に読み出される（図４：Ｓ４０１）。尚、上記アプリケーション１０２は、図２のタスク２０１に示すように、アプリケーション関数ａ（２０３）、アプリケーション関数ｂ（２０４）、アプリケーション関数ｃ（２０６）より構成される。さらに、上記アプリケーション１０２は、実行時にシステムライブラリ関数である、システムライブラリ関数ａ（２０５）を呼び出すため、例えば上記ローダーにて読み出される際に、同時に上記記憶手段１０３に格納されているシステムライブラリ１０４から、システムライブラリ関数１０７が上記一時記憶手段１０５に読み出される。

【００３７】

続いて、上記アプリケーション１０２のメイン関数であるアプリケーション関数ａ（２０３）がプログラム実行手段１０８に読み出されると、ＯＳ１３０は、アプリケーション関数ａ（２０３）に対応するタスク２０１を生成すると共に、当該タスクにリソースを割り当てる（図４：Ｓ４０２）。以後上記アプリケーション１０２は、タスク２０１として管理される。

【００３８】

さらに、上記タスクの生成時に上記ＯＳ１３０は、後述するタスク状態記憶手

段１１０、強制終了情報記憶手段１１４、及びタスク属性情報記憶手段１１７に当該タスク２０１に対応するタスクフラグ１１８を作成し、当該フラグに例えば“０”を格納する（図４：Ｓ４０３）。又、例えばプログラム実行手段は、タスク属性認識手段１１６に対して、現在タスク２０１にて実行されている関数がアプリケーション関数である旨を通知する（図４：Ｓ４０４）。当該タスク２０１にて実行されている関数がアプリケーション関数である旨の通知を受信すると、上記タスク属性認識手段１１６は、タスク属性情報記憶手段１１７のタスク２０１に対応するタスクフラグ１１８に、例えばアプリケーション関数が実行されていることを示す“１”を格納する。ここで、例えばＯＳより提供されるアプリケーションによりタスクが作成された場合、通常、メイン関数はシステムライブラリ関数であるため、上記タスク属性情報記憶手段１１７のタスク２０１に対応するタスクフラグ１１８には“０”が格納される。

【００３９】

以後、上記タスク２０１のアプリケーション関数ａ（２０３）の先頭のコマンド（命令）より順次実行される。又、必要に応じてアプリケーション関数ｂ（２０４）等の他の関数がプログラム実行手段１０８に読み出される。尚、上記コマンドが例えば１つ実行されるに際して、当該タスク（ここではタスク２０１）に強制終了命令が送信されているか否かをタスク実行判断手段１１１に問い合わせるが詳細は後述する。

【００４０】

以上が、新規アプリケーションが実行された際に行われる処理（図４における初期処理４００）である。尚、理解に供するために、プログラム実行手段１０８では、アプリケーション関数ｄ（２０８）及びアプリケーション関数ｅ（２０９）により構成されるタスク２０２も実行されるものとする。

【００４１】

次に、タスク内において実行されている関数が、強制終了禁止命令を発行した際の、タスク状態設定手段１０９の処理の詳細について説明する。

【００４２】

例えば、アプリケーション関数ａ（２０３）の処理中の所定のポイント２１３

にて、タスク２０１からＯＳ１３０に対して強制終了禁止命令２１４が送信されると、当該強制終了禁止命令２１４はタスク状態設定手段１０９にて受信される。上記強制終了禁止命令２１４を受信したタスク状態設定手段１０９は、タスク状態記憶手段１１０内の、タスク２０１に対応するフラグエリア１２１に例えば“１”を格納する。当該フラグエリア１２１に格納される“１”は、タスク２０１が現在強制終了禁止区間にある（強制終了禁止である）ということを意味するものである。

【００４３】

又、例えばアプリケーション関数ａ（２０３）の処理中の所定のポイント２１５にて、タスク２０１からＯＳ１３０に対して強制終了許可命令２１６が送信されると、当該強制終了許可命令２１６は上記タスク状態設定手段１０９にて受信される。上記強制終了許可命令２１６を受信したタスク状態設定手段１０９は、タスク状態記憶手段１１０内の、タスク２０１に対応するフラグエリア１２１に例えば“０”を格納する。当該フラグエリア１２１に格納される“０”は、現在実行されているタスク２０１が、現在強制終了禁止区間に無い（強制終了禁止ではない）ということを意味するものである。

【００４４】

以上のように、タスク状態記憶手段１１０には、各タスクの状態が強制終了禁止（フラグ“１”）であるか、強制終了可能（フラグ“０”）であるかの情報が常時格納される。以後、例えばタスク実行判断手段１１１よりタスク状態記憶手段１１０内の各タスクフラグの内容について問い合わせがあると、タスク状態設定手段１０９は当該タスクフラグに格納されている値を読み出し、上記タスク実行判断手段１１１に返信する。

【００４５】

次に、タスク内において実行されている関数が変わる際の、タスク属性認識手段１１６の処理の詳細について説明する。

【００４６】

アプリケーション関数ａ（２０３）が例えばアプリケーション関数ｂ（２０４）を呼び出し、当該タスク２０１における処理がアプリケーション関数ｂ（２０

４）に切り替わった場合、上記タスク２０１はタスク属性認識手段１１６に対して、現在タスク２０１にて実行されている関数が、アプリケーション関数である旨を通知する。ここで、上記タスク属性認識手段１１６は、タスク２０１に対応するフラグエリア１１９に“１”を格納する。

【００４７】

又、アプリケーション関数ｂ（２０４）がシステムライブラリ関数ａ（２０５）を呼び出した際にも、タスク２０１は上記タスク属性認識手段１１６に通知を行うが、この場合、タスク２０１にて実行される関数はシステムライブラリ関数であるため、上記フラグエリア１１９には“０”を格納する。尚、上記同様、システムライブラリ関数ａ（２０５）がアプリケーション関数ｃ（２０６）を呼び出した際には、上記フラグエリア１１９には“１”が格納される。

【００４８】

尚、アプリケーション関数ｃ（２０６）の処理が終了し、システムライブラリ関数ａ（２０５）に処理が戻った場合にも、上記呼び出した際と同様に、以後実行される関数（処理が戻った後の関数：ここでは“０”）に対応するフラグ“０”又は“１”がフラグエリア１１９に格納される。

【００４９】

以上のように、関数の呼び出し時、及び処理が戻った時に、タスクがタスク属性認識手段に以後実行される関数の種類を通知することで、タスク属性情報記憶手段には、現在実行されているタスクの種類が常時格納されることになる。以後、例えばタスク実行判断手段１１１よりタスク属性情報記憶手段１１７内の各タスクフラグの内容について問い合わせがあると、タスク属性認識手段１１６は当該タスクフラグに格納されている値を読み出し、上記タスク実行判断手段１１１に返信する。

【００５０】

さて、現在上記プログラム実行手段１０８においては、タスク２０１及びタスク２０２が実行されていることは上述したとおりである。ここで、タスク２０２は、上記タスク２０１に対して強制終了命令２１１を発行するタスクであるものとする。上記強制終了命令２１１が発行されるタイミング（ポイント）は、ＯＳ

１３０によるタスクの切り替えのタイミングによって異なるため、各タイミングでの詳細な処理は後述するものとし、ここでは、上記強制終了命令２１１が発行された際の、強制終了設定手段１１３の処理の詳細について説明する。

【００５１】

まず、所定のポイント（例えば図２に示したポイント２０７）にて、タスク２０１からタスク２０２に処理が切り替わったものと仮定する。当該タスクの切り替えは、従来技術にて説明したものである。

【００５２】

ここで、タスク２０２がタスク２０１に対して強制終了命令２１１を発行した場合、具体的には、例えばアプリケーション関数ｄ（２０８）にて呼び出されたアプリケーション関数ｅ（２０９）が、所定のポイント２１０にて発行した強制終了命令２１１は、強制終了設定手段１１３にて受信される。当該強制終了設定手段１１３は、上記強制終了命令２１１を受信すると、強制終了情報記憶手段１１４の、タスク２０１に対応するフラグエリア１２０に“１”を格納する。該フラグエリア１２０に格納される“１”は、フラグエリア１２０に対応するタスク２０１が強制終了命令を受けた状態であることを意味する。以後、例えばタスク実行判断手段１１１より強制終了情報記憶手段１１４内の各タスクフラグの内容について問い合わせがあると、強制終了設定手段１１３は当該タスクフラグに格納されている値を読み出し、上記タスク実行判断手段１１１に返信する。

【００５３】

以上のように、例えば他のタスクからの強制終了命令は、強制終了設定手段１１３を介して強制終了情報記憶手段１１４内の、該当するタスク（終了命令を受けるタスク）のフラグにて示される。

【００５４】

以上がタスク状態設定手段１０９、タスク属性認識手段１１６、及び強制終了設定手段１１３における各処理である。

【００５５】

次に、上述したタスク状態設定手段１０９、タスク属性認識手段１１６、及び強制終了設定手段１１３の各処理を踏まえた上で、新規アプリケーションが実行

された後に行われる処理の詳細について説明する。

【００５６】

図４に示した初期処理４００を完了した後、プログラム実行手段１０８は、上記タスク２０１のアプリケーション関数ａ（２０３）の先頭のコマンド（命令）を順次実行する（図４：Ｓ４０５）。ここで、上記プログラム実行手段１０８は、コマンドが例えば１つ実行されるに際して、当該タスク（ここではタスク２０１）に強制終了命令が送信されているか否かを、例えばＯＳ１３０の機能として提供されているタスク実行判断手段１１１に問い合わせる。上記タスク実行判断手段１１１は、上記問い合わせを受け取ると、上記強制終了設定手段１１３に対して強制終了情報記憶手段１１４のタスク２０１フラグ（１２０）の内容を問い合わせる。尚、当該タスクフラグの内容は上述したとおりである。

【００５７】

ここで、タスク２０１フラグ（１２０）が“１”ではないと上記強制終了設定手段１１３より返信があった場合、上記タスク２０１は強制終了命令を発行されていないことを示し、上記タスク実行判定手段１１１はプログラム実行手段１０８に対しその旨を通知することで、当該プログラム実行手段１０８は次のコマンドを実行する（図４：Ｓ４０６Ｎｏ→Ｓ４０５）。尚、次のコマンドが例えばアプリケーション関数ｂ（２０３）を呼び出すコマンドである場合等には、タスク属性情報記憶手段１１７のタスク２０１フラグ（１１９）の書き換えを随時行うのは上述したとおりである。

【００５８】

又、タスク２０１フラグ（１２０）が“１”であると上記強制終了設定手段１１３より返信があった場合、上記タスク２０１は強制終了命令を発行されていることを示し、さらに上記タスク実行判断手段１１１は、タスク状態設定手段１０９に対してタスク状態記憶手段１１０のタスク２０１フラグ（１２１）の内容を問い合わせる（図４：Ｓ４０６Ｙｅｓ→Ｓ４０７）。

【００５９】

ここで、タスク２０１フラグ（１２１）が“１”ではないと上記タスク状態設定手段１０９より返信があった場合、上記タスク２０１は強制終了禁止区間でな

い、即ち、当該タスク２０１を終了してもシステム全体に影響を及ぼさない（共有リソースに対して操作を行っていない）ことを表している。従って、上記タスク実行判定手段１１１はプログラム実行手段１０８に対しその旨を通知することで、当該プログラム実行手段１０８は、実行している関数（例えばアプリケーション関数ｃ（２０６））及び当該関数を呼び出した関数（例えばアプリケーション関数ａ（２０３）、アプリケーション関数ｂ（２０４）、システムライブラリ関数ａ（２０５））をすべて終了し、タスク２０１を終了する（図４：Ｓ４０７Ｎｏ→Ｓ４０８→Ｅｎｄ）。

【００６０】

又、タスク２０１フラグ（１２１）が“１”であると上記タスク状態設定手段１０９より返信があった場合、上記タスク２０１は強制終了禁止区間であり、即ち、当該タスク２０１を終了するとシステム全体に影響を及ぼす可能性がある（共有リソースに対して操作を行っている等）ことを表している。この場合にはさらに、上記タスク実行判断手段１１１を構成するカレント関数実行判定手段１１２が、タスク属性認識手段１１６に対して、タスク属性情報記憶手段１１７のタスク２０１フラグ（１１９）の内容を問い合わせる（図４：Ｓ４０７Ｙｅｓ→Ｓ４０９）。

【００６１】

ここで、タスク２０１フラグ（１１９）が“１”ではない（“０”である）と上記タスク属性認識手段１１６より返信があった場合、上記タスク２０１は現在システムライブラリ関数を実行していることを意味し、即ち、当該システムライブラリ関数を終了するとシステム全体に影響を及ぼす可能性がある（共有リソースに対して操作を行っている等）ため、当該システムライブラリ関数の実行を強制終了させること無く、次のコマンドの処理に移る（図４：Ｓ４０９Ｎｏ→Ｓ４０５）。

【００６２】

又、タスク２０１フラグ（１１９）が“１”であると上記タスク属性認識手段１１６より返信があった場合、上記タスク２０１は現在アプリケーション関数を実行していることを意味し、当該アプリケーション関数を終了してもシステム全

体に影響を及ぼす可能性が無い（低い）ため、当該実行している関数を終了する（図４：Ｓ４０９Ｙｅｓ→Ｓ４１０）。尚、当該実行している関数がメイン関数であった場合には、強制終了するとシステム全体に影響を及ぼす可能性があるためタスク２０１を終了する（図４：Ｓ４１０→Ｓ４０５→Ｅｎｄ）。

【００６３】

以上のように、本実施の形態においては、強制終了情報記憶手段に格納されている強制終了情報（強制終了命令受信の有無）と、タスク状態記憶手段に格納されているタスク状態（強制終了禁止区間であるか否か）に加えて、タスク属性情報記憶手段に格納されている、現在のタスクにて実行されている関数の属性情報（アプリケーション関数か、システムライブラリ関数か）を参照・判定している。従って、強制終了禁止区間であっても、アプリケーション関数を実行している場合には、強制終了（当該アプリケーション関数の中断）を可能とし、一方でシステムに障害をもたらす可能性のあるシステムライブラリ関数の実行時には、当該タスクを強制終了させないといった強制終了についての柔軟な制御が可能である。

【００６４】

続いて、図３、図４を用いて、強制終了命令を受けた各ポイントによって異なる処理の詳細を説明する。尚、当該各処理は、上記図４のフローチャートを場合分けして表すものであって、内容が異なるものではない。

【００６５】

まず、強制終了命令をアプリケーション関数ａの強制終了禁止区間設定前に受けた場合（強制終了命令３０１、３０９）、アプリケーション関数ａを終了し、即ちタスク２０１を終了する（図４：Ｓ４０７Ｎｏ→Ｓ４０８）。

【００６６】

強制終了命令をアプリケーション関数ａ実行時の強制終了禁止区間設定後に受けた場合（強制終了命令３０２、３０８）、アプリケーション関数ａを終了し、即ちタスク２０１を終了する（図４：Ｓ４０７Ｙｅｓ→Ｓ４０９→Ｓ４１０→Ｓ４０５→Ｅｎｄ）。

【００６７】

強制終了命令をアプリケーション関数ｂ実行時に受けた場合（強制終了命令３０３、３０７）、アプリケーション関数ｂを終了し、アプリケーション関数ａに処理が戻るが、アプリケーション関数ａも直ちに終了することになり、即ちタスク２０１を終了する（図４：Ｓ４０７Ｙｅｓ→Ｓ４０９Ｙｅｓ→Ｓ４１０→Ｓ４０５を２回繰り返してＥｎｄへ）。

【００６８】

強制終了命令をシステムライブラリ関数ａ実行時に受けた場合（強制終了命令３０４、（強制終了命令３０６を含む））、当該システムライブラリ関数ａの実行を終了せずに、アプリケーション関数ｃを呼び出す。ここで、当該アプリケーション関数ｃは直ちに終了するため（後述する強制終了命令３０５のケース）再度ポイント３２０よりポイント３２１までシステムライブラリ関数ａを実行し、アプリケーション関数ｂに処理を戻す。続く当該アプリケーション関数ｂ及びアプリケーション関数ａは直ちに終了し、即ちタスク２０１が終了する（図４：Ｓ４０７Ｙｅｓ→Ｓ４０９Ｎｏ→Ｓ４０５→Ｓ４０６Ｙｅｓ→Ｓ４０７Ｙｅｓ→Ｓ４０９Ｙｅｓ→Ｓ４１０（アプリケーション関数ｃ終了）→Ｓ４０５→・・・→４０９Ｎｏ→Ｓ４０５・・・（システムライブラリ関数ａ終了後）→Ｓ４０９Ｙｅｓ→Ｓ４１０（アプリケーション関数ｂ終了）→Ｓ４０５→・・・→Ｓ４１０（アプリケーション関数ａ終了）→Ｅｎｄへ）。

【００６９】

強制終了命令をアプリケーション関数ｃ実行時に受けた場合（強制終了命令３０５）、アプリケーション関数ｃを終了し、システムライブラリ関数ａのポイント３２０に処理が戻り、ポイント３２１まで処理が実行された後、アプリケーション関数ｂ、アプリケーションａに処理が戻ると共に直ちに処理が終了し、即ちタスク２０１が終了する（図４：Ｓ４０７Ｙｅｓ→Ｓ４０９Ｙｅｓ→Ｓ４１０（アプリケーション関数ｃ終了）→Ｓ４０５→・・・→Ｓ４０９Ｎｏ→Ｓ４０５→・・・→４０９Ｎｏ→Ｓ４０５・・・（システムライブラリ関数ａ終了後）→Ｓ４０９Ｙｅｓ→Ｓ４１０（アプリケーション関数ｂ終了）→Ｓ４０５→・・・→Ｓ４１０（アプリケーション関数ａ終了）→Ｅｎｄへ）。

【００７０】

以上が本実施の形態１における処理である。但し、本実施の形態１において、強制終了禁止区間を設定可能である場合の処理について説明したが、必ずしも強制終了禁止区間の判定は必要ではない。即ち、強制終了禁止区間であるか否かの判定をすることなく、現在実行されている関数の属性情報のみに基づいて強制終了の可否を判定してもよい。この場合には、図４に示したＳ４０７及びＳ４０８の処理を除くのみでよい。

【００７１】

又、上記関数の属性として、アプリケーション関数とシステムライブラリ関数を例にあげたが、特に上記２つに限定する必要は無く、使用する言語によってさらに異なった関数の場合分けが可能である場合には、当該異なる関数の属性の情報を判断基準として加えてもよい。

【００７２】

〔実施の形態２〕

次に、実施の形態２では、図１を用いてタスク属性記憶手段のタスクフラグに情報を格納する際の処理について説明する。

【００７３】

上記実施の形態１では、プログラム実行手段１０８が現在実行している関数の属性（アプリケーション関数、システムライブラリ関数）を、直接タスク属性認識手段１１６に通知することにより、タスク属性情報記憶手段１１７の各タスクフラグに当該属性を格納していた。

【００７４】

即ち、プログラム実行手段１０８は、現在実行している関数が変更になった場合、当該変更になった関数名（これから実行する関数の名称）を上記タスク属性認識手段１１６に送信する。当該タスク属性認識手段１１６は、上記関数名を受信すると、一時記憶手段１０５に格納されているアプリケーション関数１０６やシステムライブラリ関数１０７を参照することで、当該関数がアプリケーション関数かシステムライブラリ関数かを判断する。上記関数の属性を判断したタスク属性認識手段１１６は、当該関数の種類を該当するタスクフラグに格納する。

【００７５】

以上により、直接プログラム実行手段より関数属性をタスク属性認識手段１１６に送信できない場合であっても、タスク属性情報記憶手段は、該当するタスクフラグに現在実行されている関数の属性を格納することが可能になる。

【００７６】

ところで、上記実施の形態１、及び実施の形態２においては、強制終了を実行する判断の基準に、タスク内で現在実行されている関数の属性を用いている。しかしながら、上記判断の基準を必ずしも関数の属性のみに限定する必要は無い。即ち、マルチタスクＯＳでは、ＯＳの機能の一部を独立したタスクとして実行することが可能であり、これをシステムタスク（システムプロセス、システムスレッド等）と呼ぶ。又、これに対して、ユーザーが起動したタスクがアプリケーションタスク（ユーザープロセス、アプリケーションスレッド等）である。

【００７７】

上記システムタスクとアプリケーションタスクは、タスク属性ということが可能であり、当該２つのタスク属性を比較した場合には以下のようなことが言える。

【００７８】

即ち、上記システムタスクは、ＯＳの機能として提供されているために信頼性（安全性）が高く、一方上記アプリケーションタスクは悪意を持った処理を行う可能性があり、上記システムタスクに比較して信頼性が低い。さらに、上記システムタスクは強制終了されるとシステムに障害をもたらす可能性が高いが、上記アプリケーションタスクは当該タスクがシステムライブラリ関数を利用しているものであったとしても、一般的に強制終了された場合の影響は当該アプリケーションタスク内で完結するものである。

【００７９】

従って、以上のことを考慮すると、上記判断の基準を関数の属性ではなく、タスクの属性で判断することができるといえる。即ち、上記タスク属性情報記憶手段が記憶するタスクフラグに、関数の属性ではなくタスクの属性を格納するのである。当該格納する処理は、タスクが起動した時の上記初期処理時に、プログラム実行手段がタスク属性認識手段に対し一回だけ行うのみでよい。格納する値は

、例えばシステムタスクの場合には“０”であり、アプリケーションタスクの場合には“１”である。以後、タスク属性情報記憶手段内の該当するタスクフラグのチェックは同様である。

【００８０】

以上のように、関数の属性に替えてタスクの属性を管理し、当該タスクに対する強制終了の処理の実行を判断することで、強制終了についての柔軟な制御が可能である。

【００８１】

【発明の効果】

以上のように、本発明によれば、関数の呼び出し時、及び処理が戻った時に、タスクがタスク属性認識手段に以後実行される関数の種類を通知することで、タスク属性情報記憶手段には、現在実行されているタスクの種類が常時格納される構成としている。このため、タスク属性情報記憶手段に格納されている、現在のタスクにて実行されている関数の属性情報を参照・判定可能となり、従って、強制終了禁止区間であっても、所定の関数を実行している場合には、強制終了可能とし、一方でシステムに障害をもたらす可能性のある別の関数の実行時には、当該タスクを強制終了させないといった強制終了についての柔軟な制御が可能である。

【００８２】

又、タスク属性情報記憶手段が、関数の属性に代えてタスクの属性を格納することにより、タスクの属性に基づいた強制終了の制御が可能である。

【図面の簡単な説明】

【図１】

本発明に係るデジタル機器における概略機能ブロック図。

【図２】

本発明に係るタスク間にわたる強制終了命令発行時のイメージ図。

【図３】

各関数実行時における強制終了命令受信のポイントを示す図。

【図４】

タスク実行時の強制終了命令処理に関するフローチャート。

【図５】

従来技術におけるプロセス、スレッド及びＯＳのイメージを示す図。

【図６】

タスク内の関数構造の一例を示す図。

【図７】

従来技術におけるタスク間にわたる強制終了命令発行時のイメージ図。

【符号の説明】

１００　デジタル機器

１０１　ネットワーク

１０２　アプリケーション

１０３　記憶手段

１０４　システムライブラリ

１０５　一時記憶手段

１０６　アプリケーション関数

１０７　システムライブラリ関数

１０８　プログラム実行手段

１０９　タスク状態設定手段

１１０　タスク状態記憶手段

１１１　タスク実行判断手段

１１２　カレント関数実行判定手段

１１３　強制終了設定手段

１１４　強制終了情報記憶手段

１１６　タスク属性認識手段

１１７　タスク属性情報記憶手段

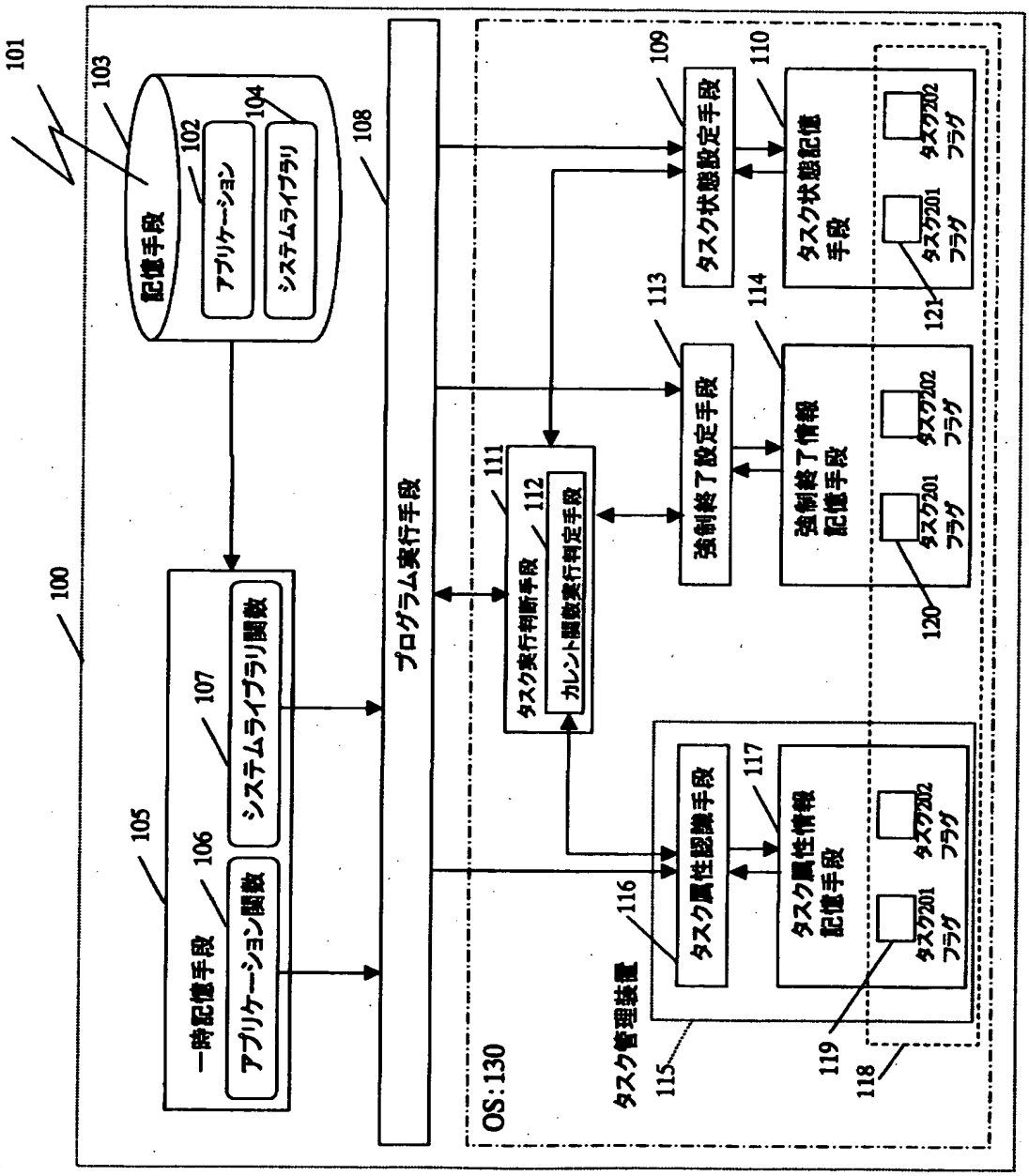
１１８　タスクフラグ

１１９、１２０、１２１　フラグエリア

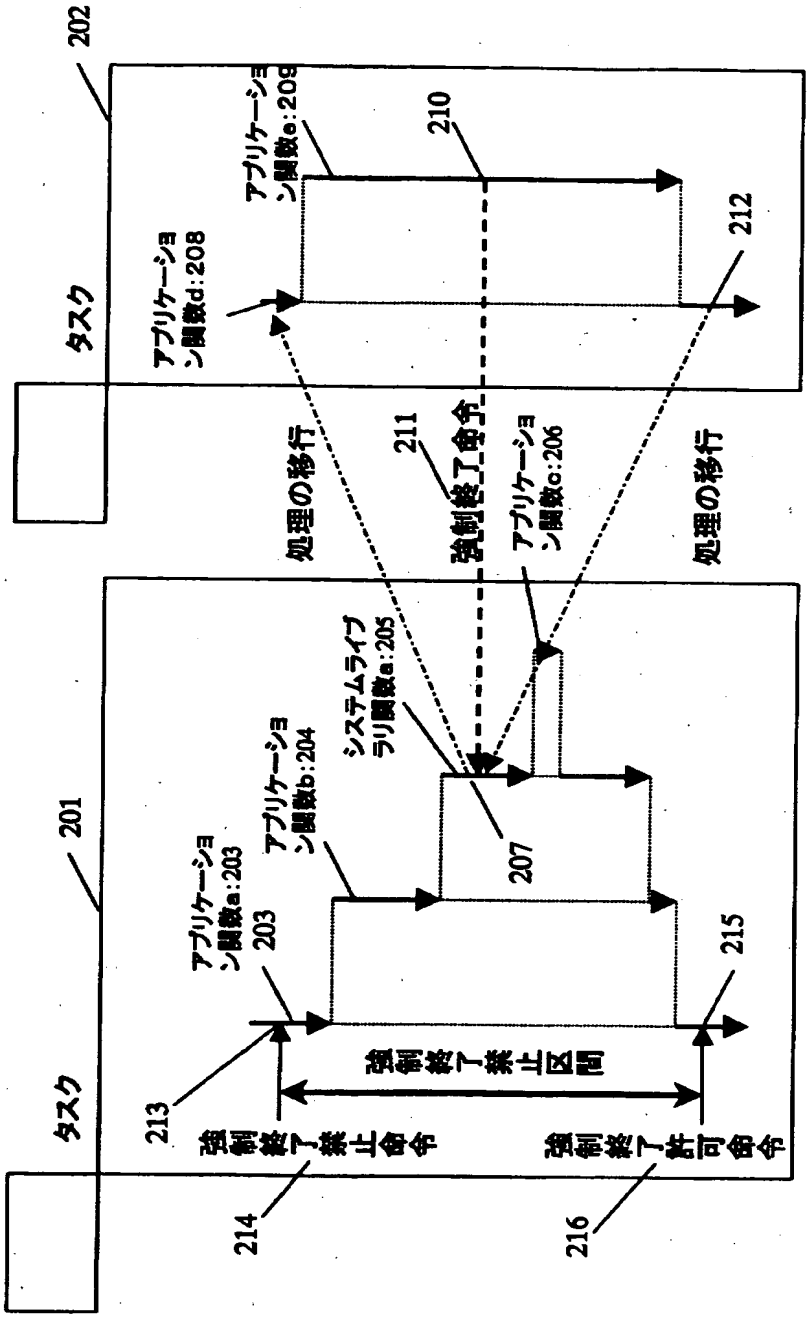
１３０　ＯＳ（ＯＳ機能）

【書類名】　　図面

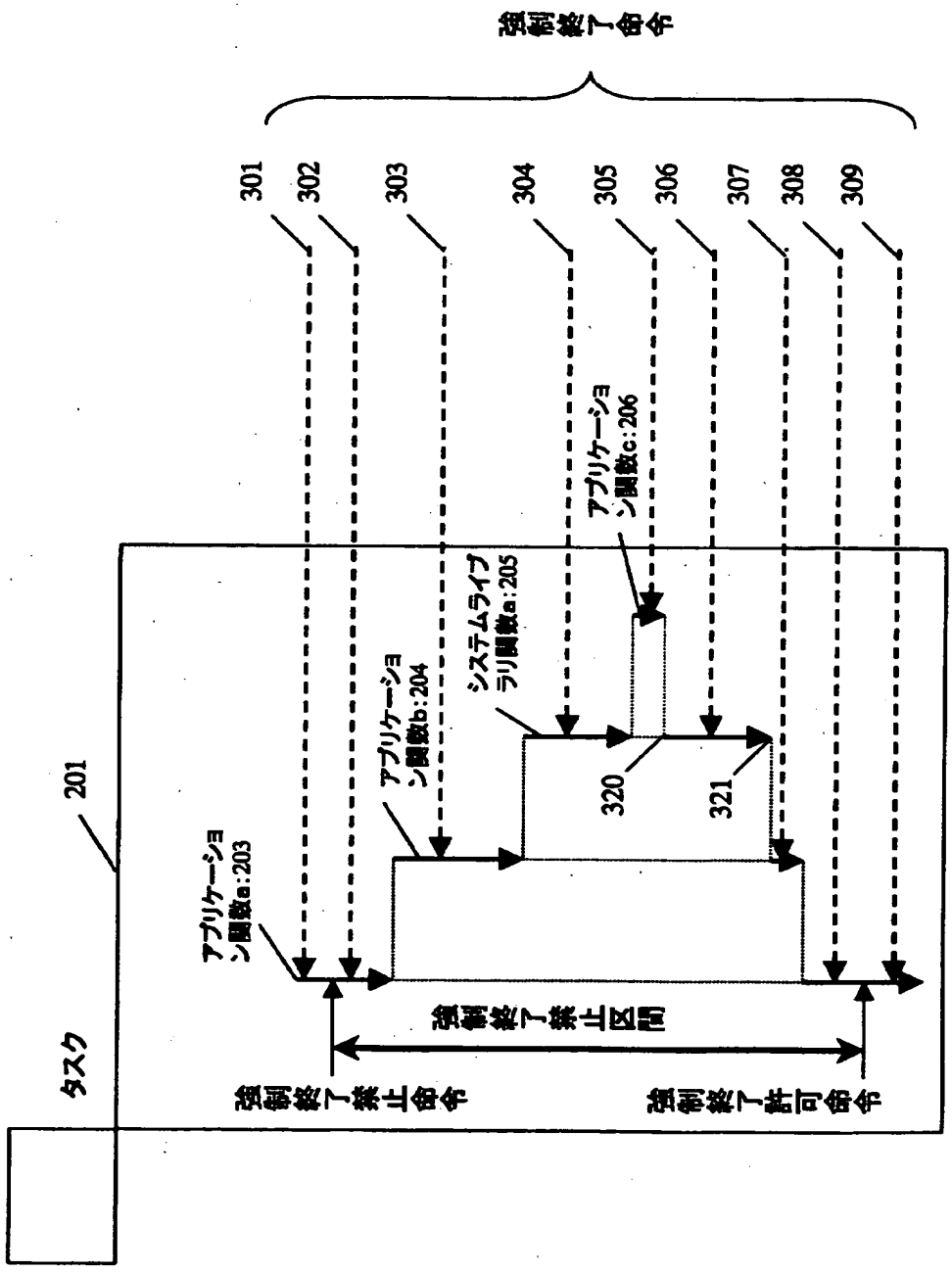
【図１】


101
103
記憶手段
102
アプリケーション
104
システムライブラリ
108
プログラム実行手段
109
タスク状態設定手段
110
タスク状態記憶手段
タスク202フラグ
タスク201フラグ
121
113
強制終了設定手段
114
強制終了情報記憶手段
タスク202フラグ
タスク201フラグ
120
111
タスク実行判断手段
112
カレント関数実行判定手段
100
一時記憶手段
105
107
システムライブラリ関数
106
アプリケーション関数
タスク管理装置
116
タスク属性認識手段
117
タスク属性情報記憶手段
タスク202フラグ
タスク201フラグ
119
115
OS:130
118

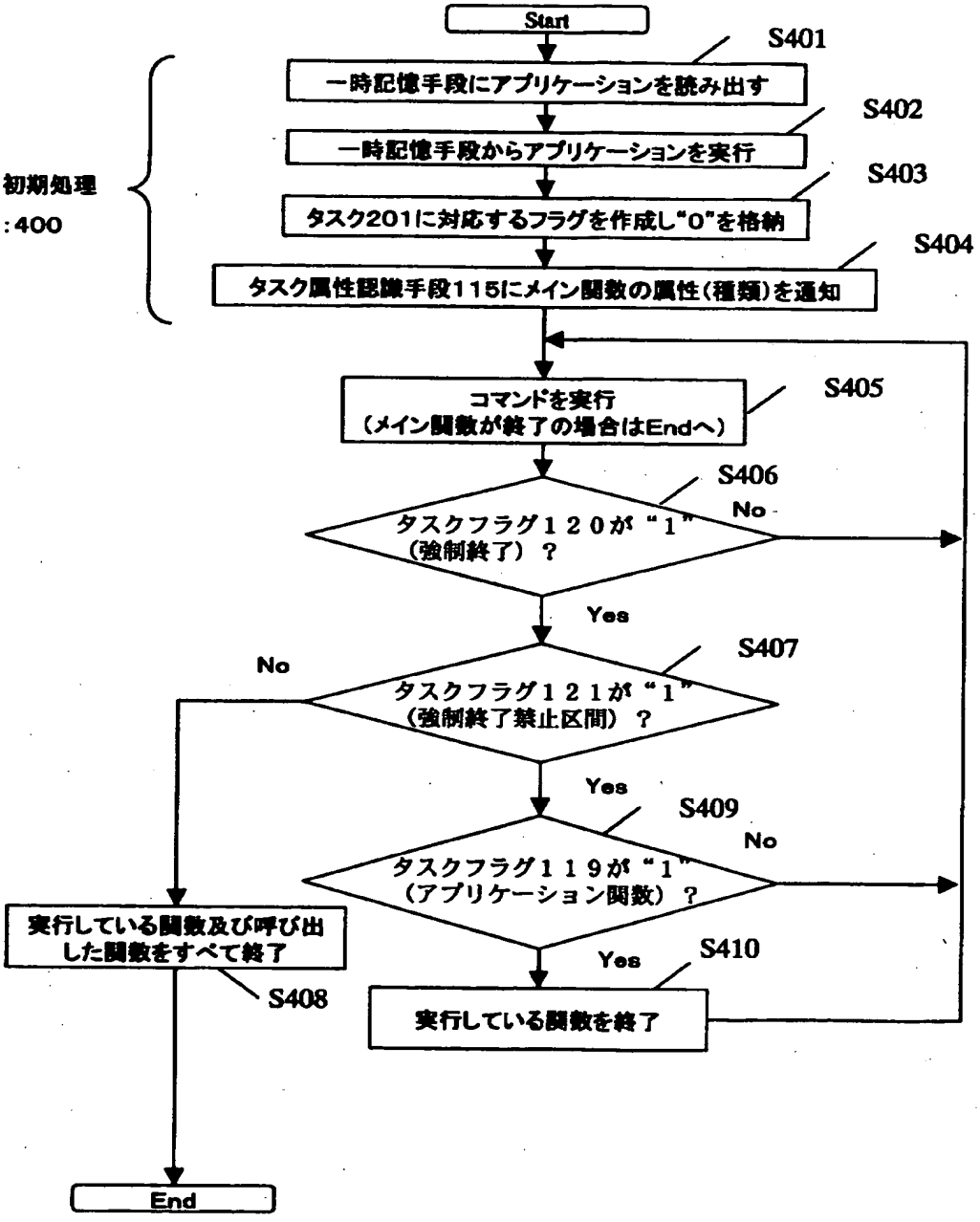
【図２】


202
タスク
アプリケーション関数d:208
アプリケーション関数e:209
210
212
処理の移行
211
強制終了命令
アプリケーション関数c:206
処理の移行
201
タスク
アプリケーション関数a:203
203
アプリケーション関数b:204
システムライブラリ関数a:205
207
215
213
強制終了禁止区間
強制終了禁止命令
強制終了許可命令
214
216

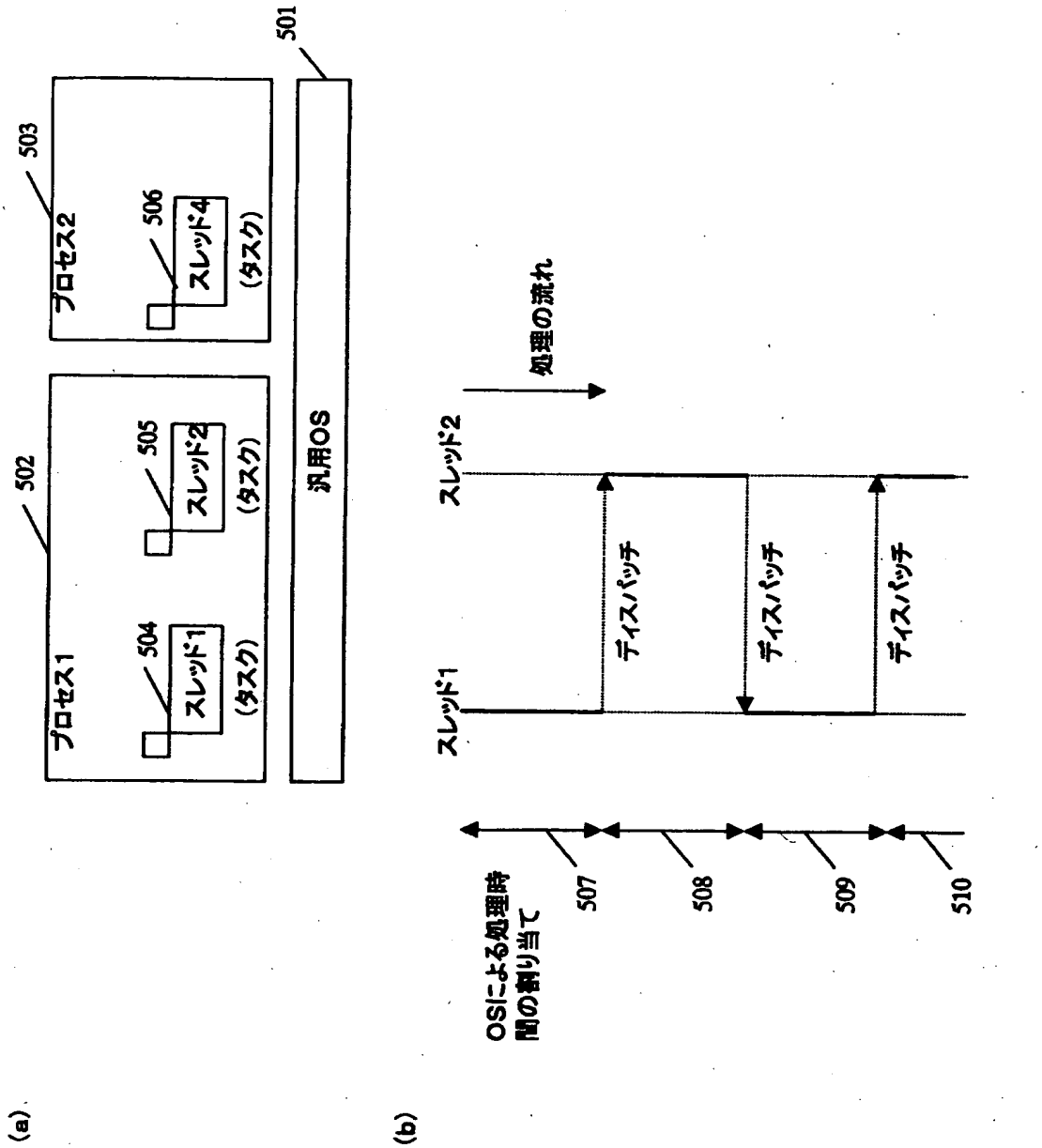
【図３】


301
302
303
304
305
306
307
308
309
アプリケーション関数c:206
システムライブラリ関数a:205
アプリケーション関数b:204
アプリケーション関数a:203
320
321
201
タスク

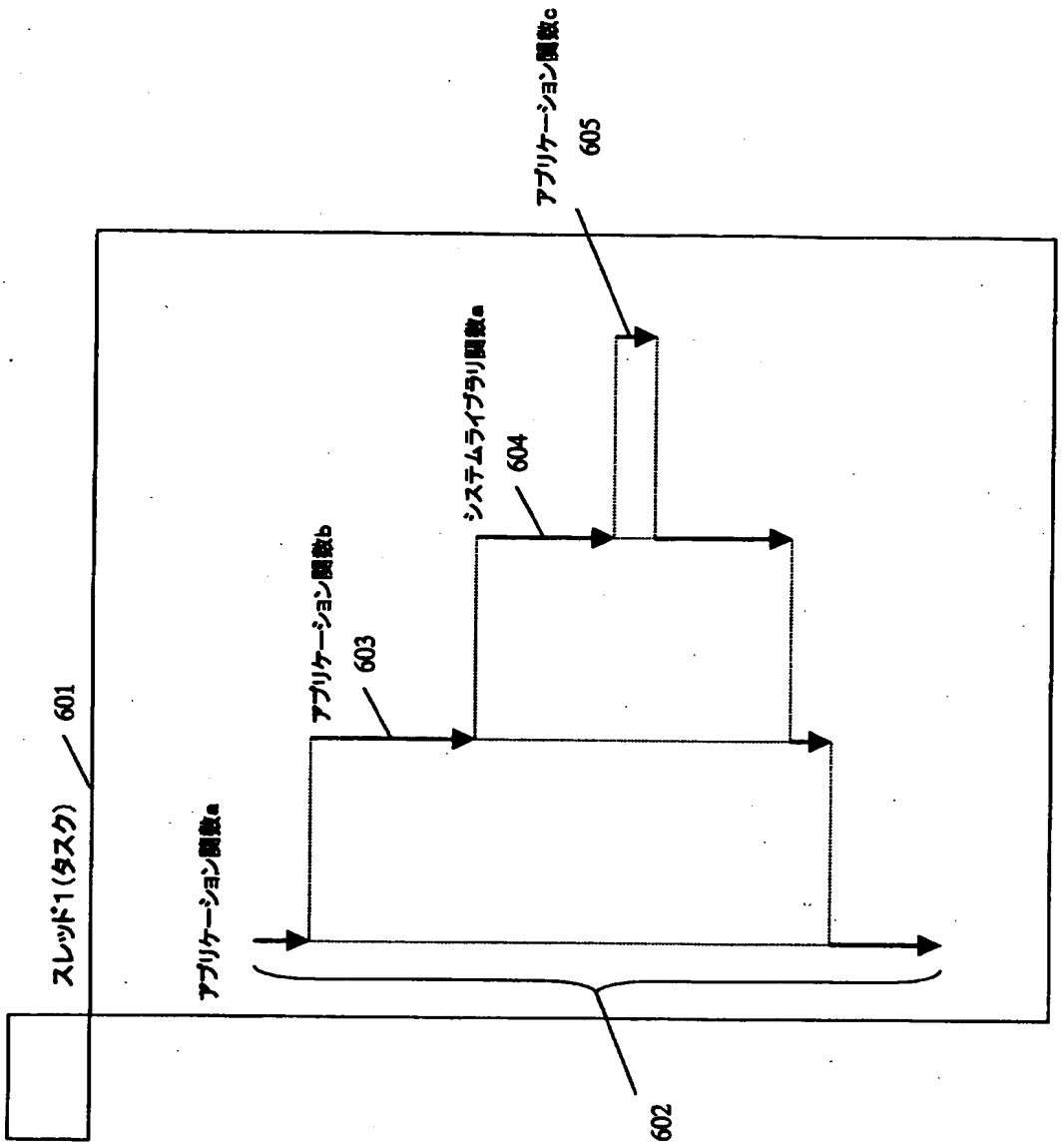
【図４】


Start
S401
一時記憶手段にアプリケーションを読み出す
S402
一時記憶手段からアプリケーションを実行
S403
タスク201に対応するフラグを作成し"0"を格納
S404
タスク属性認識手段115にメイン関数の属性(種類)を通知
初期処理
:400
コマンドを実行
(メイン関数が終了の場合はEndへ)
S405
S406
タスクフラグ１２０が“１”
(強制終了)？
No
Yes
S407
No
タスクフラグ１２１が“１”
(強制終了禁止区間)？
Yes
S409
No
タスクフラグ１１９が“１”
(アプリケーション関数)？
実行している関数及び呼び出した関数をすべて終了
S408
Yes
S410
実行している関数を終了
End

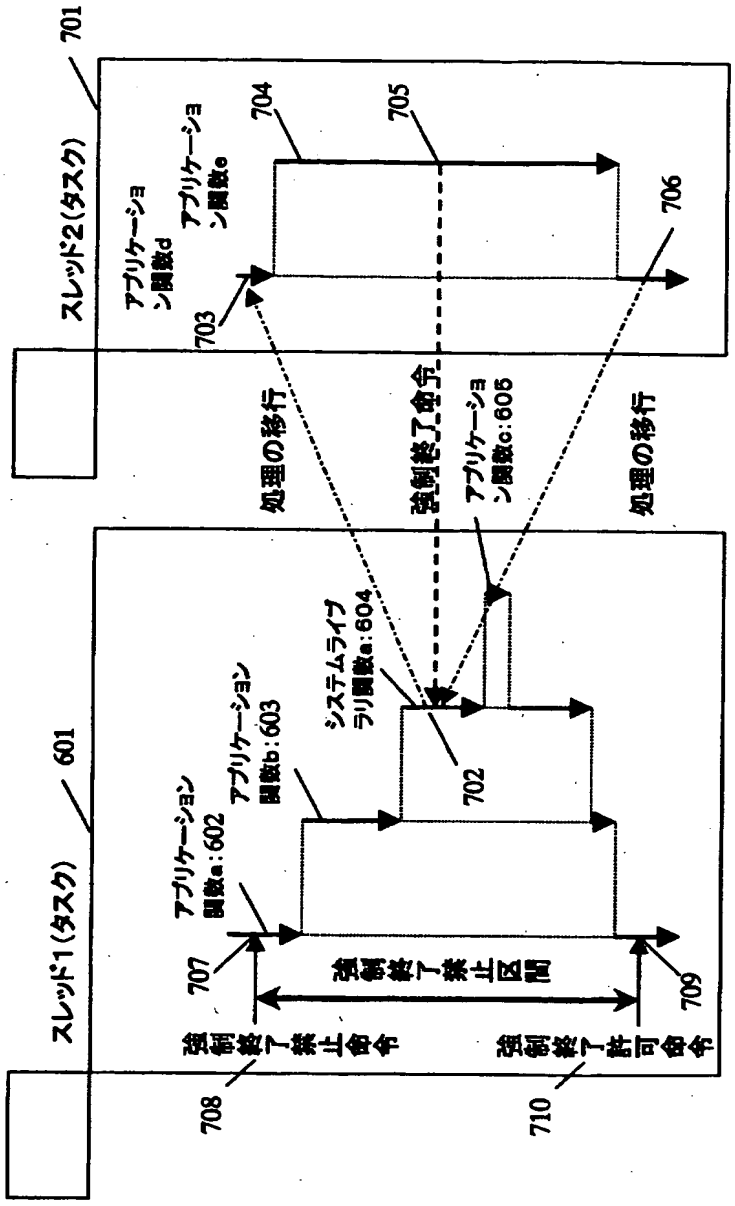
【図５】


501
503
プロセス2
506
スレッド4
(タスク)
502
プロセス1
505
スレッド2
(タスク)
504
スレッド1
(タスク)
汎用OS
(a)
処理の流れ
スレッド2
スレッド1
ディスパッチ
ディスパッチ
ディスパッチ
OSによる処理時間の割り当て
507
508
509
510
(b)

【図６】


アプリケーション関数c
605
システムライブラリ関数a
604
アプリケーション関数b
603
601
スレッド1(タスク)
アプリケーション関数a
602

【図７】


701
スレッド2(タスク)
アプリケーション関数d
アプリケーション関数e
704
705
706
703
処理の移行
強制終了命令
アプリケーション関数c:605
処理の移行
601
スレッド1(タスク)
システムライブラリ関数a:604
アプリケーション関数b:603
アプリケーション関数a:602
702
707
領域獲得禁止区間
709
領域獲得禁止命令
領域獲得許可命令
708
710

【書類名】　要約書

【要約】

【課題】　システムに障害をもたらす可能性のあるタスクの操作中には、当該タスクを強制終了させないタスク管理方法を提供する。

【解決手段】　プログラム実行手段は、実行されている関数の属性に関する情報を送信すると共に、当該関数の属するタスクに対する強制終了の実行を問い合わせる。又、タスク事項判断手段は、実行されている関数の属性に基づいて上記強制終了の実行を判定する。さらに、タスク属性認識手段は、実行されている関数の属性をタスク属性情報記憶手段に格納し、タスク実行手段からの関数の属性の問い合わせに対して、格納した関数の属性を返信する。

【選択図】　図１

出　願　人　履　歴　情　報

識別番号　　〔０００００５８２１〕

１．変更年月日　　１９９０年　８月２８日
［変更理由］　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社